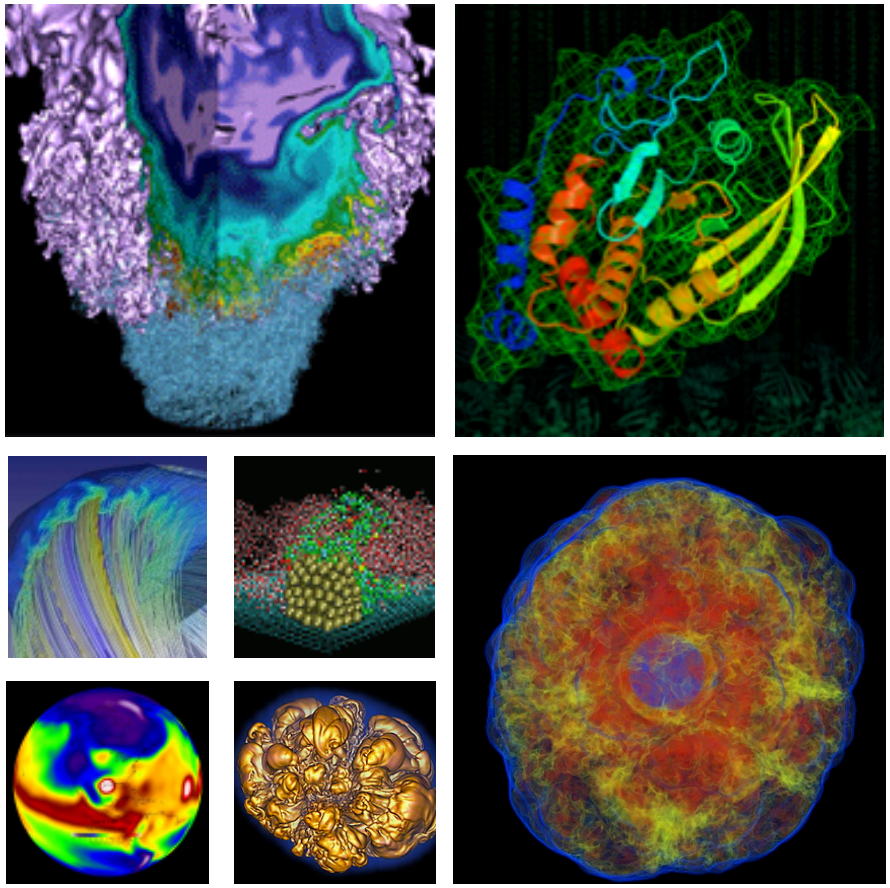
# Getting Started at NERSC




## Daniel Udwary

**NERSC Data Science Engagement Group**


**December 16, 2015**

U.S. DEPARTMENT OF ENERGY | Office of Science

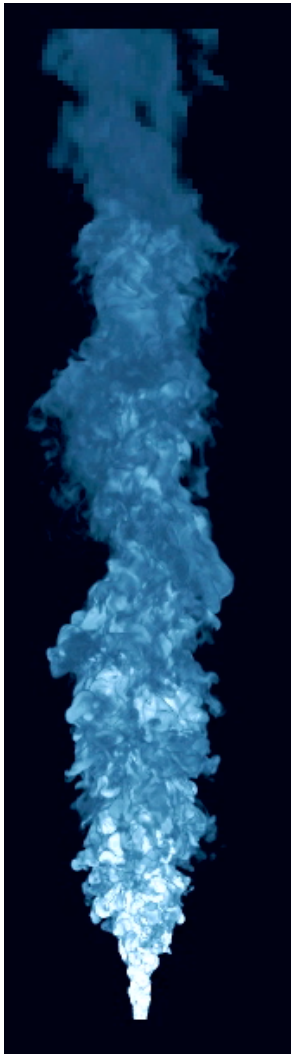
BERKELEY LAB

# Purpose

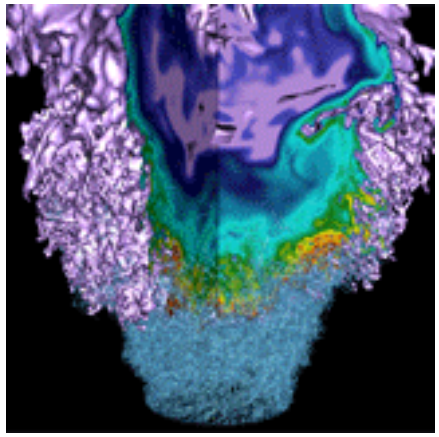
- **This presentation will help you get familiar with NERSC and its facilities**
  - Practical information
  - Introduction to terms and acronyms
- **This is not a programming tutorial**
  - But you will learn how to get help and what kind of help is available
  - We can give presentations on programming languages and parallel libraries – just ask

# Outline

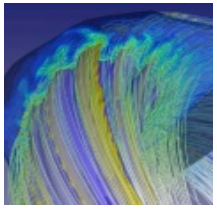
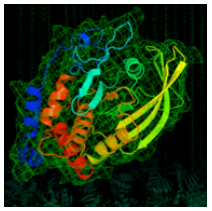
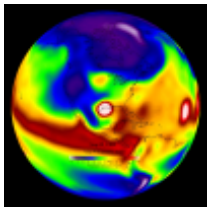
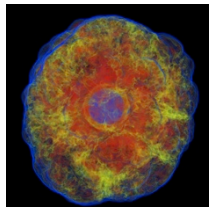
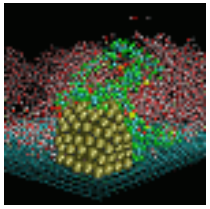
- Computing Resources
- How to Get Help
- Storage Resources
- Connecting to NERSC systems
- Modules
- Running and Monitoring Jobs



# Computing Resources



NERSC

# Current NERSC Systems

## Large-Scale Computing Systems

**Edison (NERSC-7): Cray Cascade**

- Over 200 Tflop/s on applications, 2 Pflop/s peak

**Cori (NERSC-8):**

- Currently operational, in testing

**NERSC-9 in planning**



**Midrange**

>140 Tflops total

**PDSF (HEP/NP)**

- ~1K core cluster

**GenePool (JGI)**

- ~5K core clusters
- 7.1 PB GPFS File System

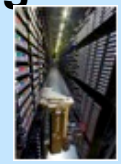
**NERSC Global Filesystem (NGF)**

Uses IBM's GPFS

- 8.5 PB capacity
- 15GB/s of bandwidth

**HPSS Archival Storage**

- 240 PB capacity
- 5 Tape libraries
- 200 TB disk cache

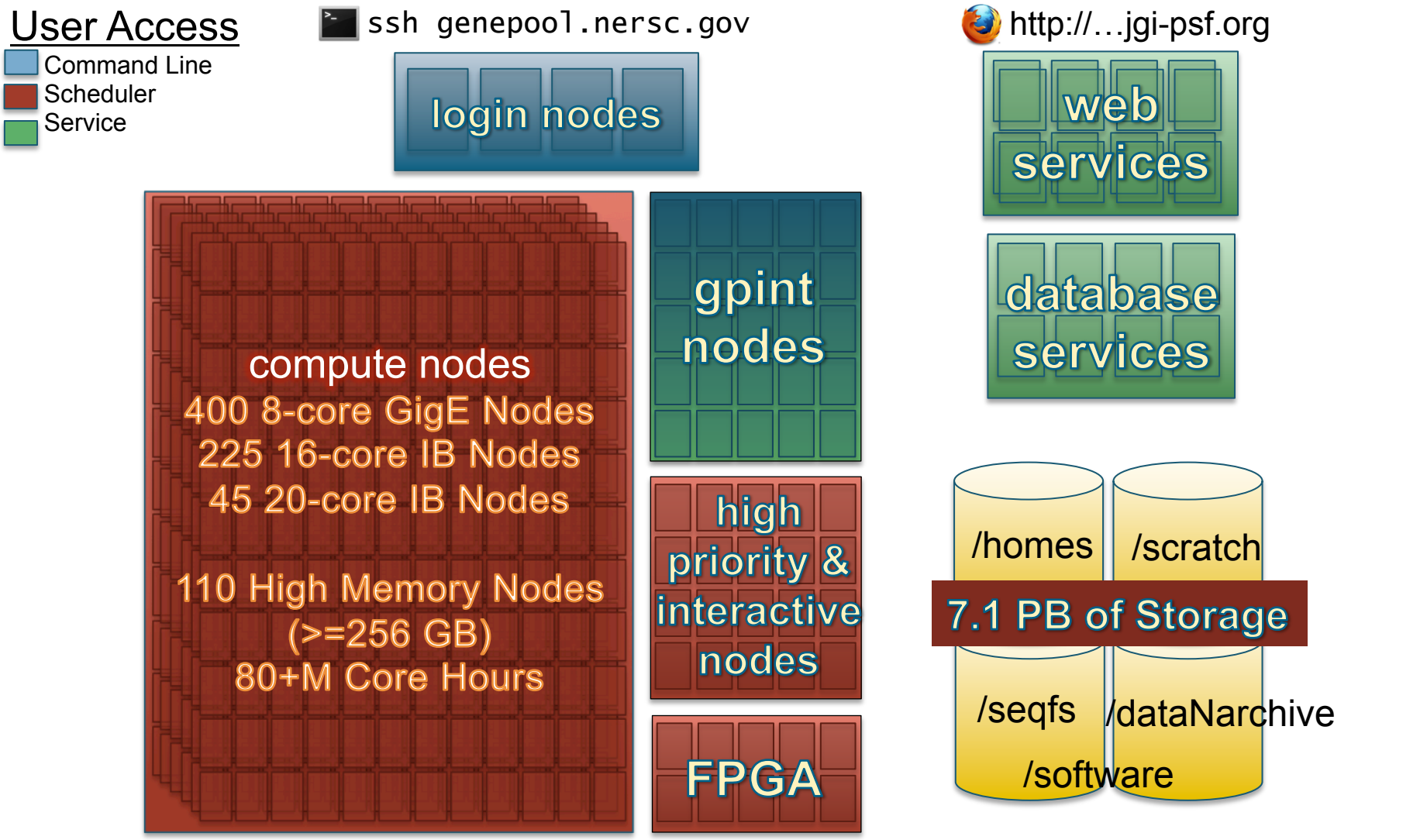


**Analytics & Testbeds**

**Babbage**

Xeon Phi

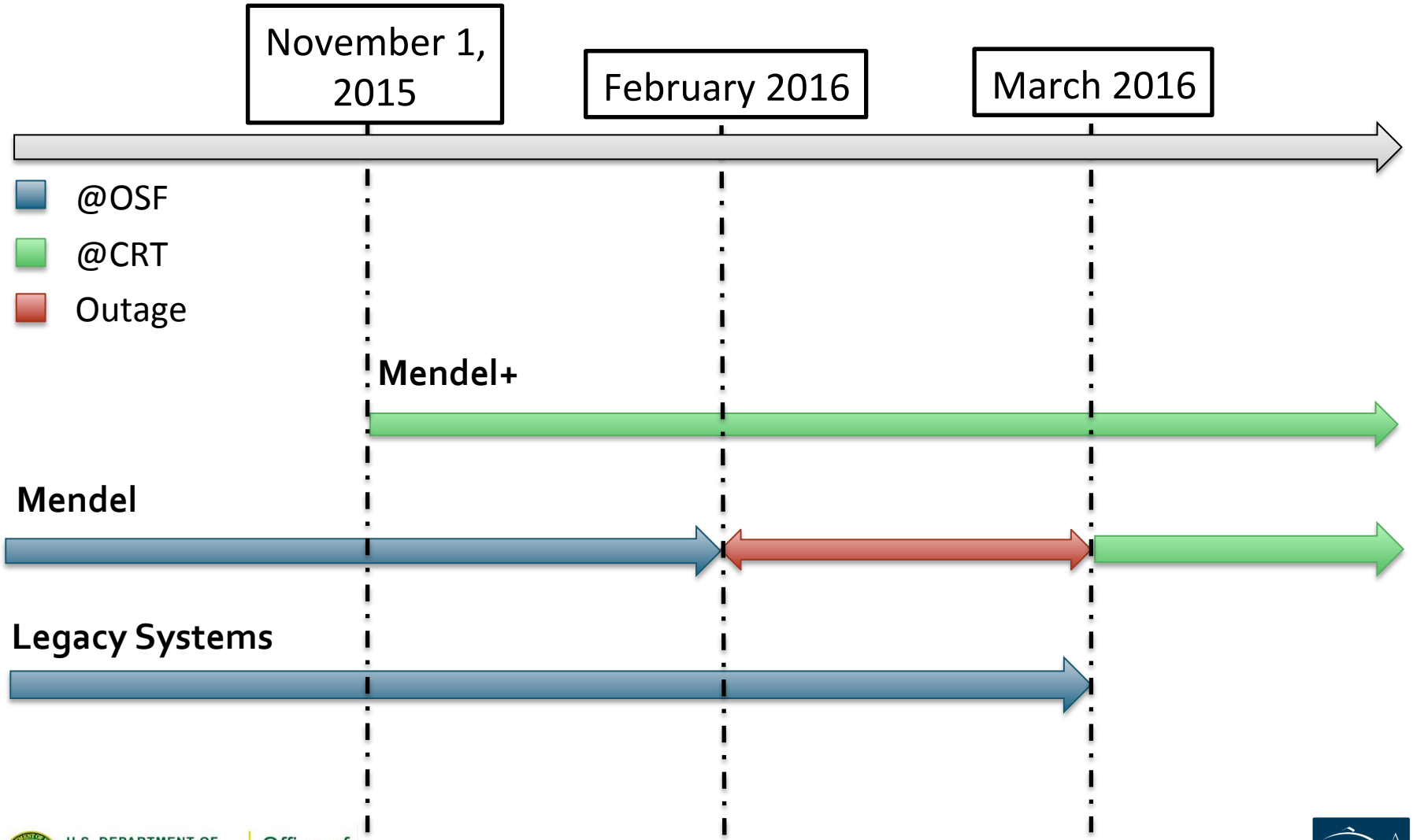
# Structure of the Genepool system



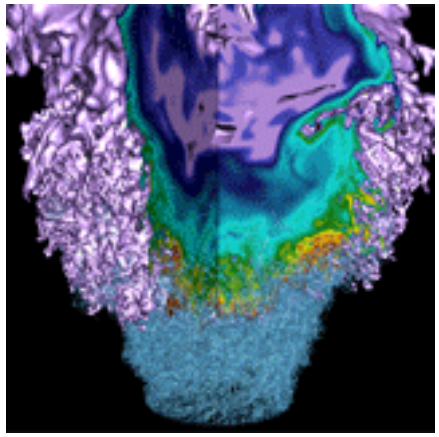
# NERSC is moving to a new building

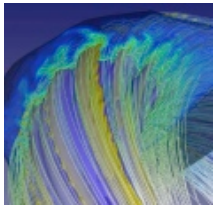
- **All systems must move from Oakland to Berkeley**

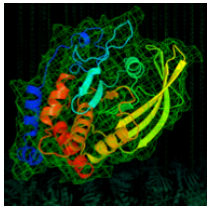
# Move Schedule Impact - Compute

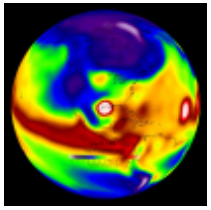


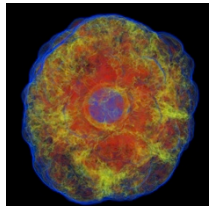
# How to Get Help

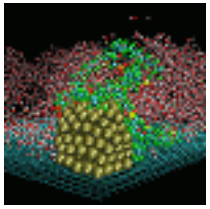












# NERSC Services

- **NERSC's emphasis is on enabling scientific discovery**
- **User-oriented systems and services**
  - We think this is what sets NERSC apart from other centers
- **Help Desk / Consulting**
  - Immediate direct access to consulting staff that includes many Ph.Ds
- **User group (NUG) has tremendous influence**
  - Monthly teleconferences & yearly meetings
- **Requirement-gathering workshops with top scientists**
  - One each for the six DOE Program Offices in the Office of Science
  - http://www.nersc.gov/science/requirements-workshops/
- **Ask, and we'll do whatever we can to fulfill your request**

# Your JGI Consultants

Kjiersten Fagnan, PhD
Applied Math

Dan Udwary, PhD
Bioorganic chemistry,
Bioinformatics


Additional consultant,
to be hired soon

Office hours are W & Th, 10-12.
Stop by 400-413 if you have questions!
consult@nersc.gov

# Where to get started on getting help

- **NERSC Genepool webpage**
  - https://www.nersc.gov/users/computational-systems/genepool/
- **Online Helpdesk – help.nersc.gov**
  - Create and monitor trouble tickets
- **NERSC Information management (NIM) webpage**
  - https://nim.nersc.gov/ - change NERSC password
- **my.nersc.gov**
  - More information on your account and usage
- **Consulting line – 1-800-66-NERSC (menu option 3)**
  - Talk to a real live consultant 8-5, M-F

# Connecting to Genepool

- **ssh genepool.nersc.gov**
  - Will take you to the least-utilized login node
  - ssh in MacOS, Linux. Putty commonly used in Windows
- **ssh gpint[xxx].nersc.gov**
  - If your group owns its own interactive node

- **Use NX for graphical (X-Windows) applications**
  - https://www.nersc.gov/users/connecting-to-nersc/using-nx/

# Passwords and Login Failures

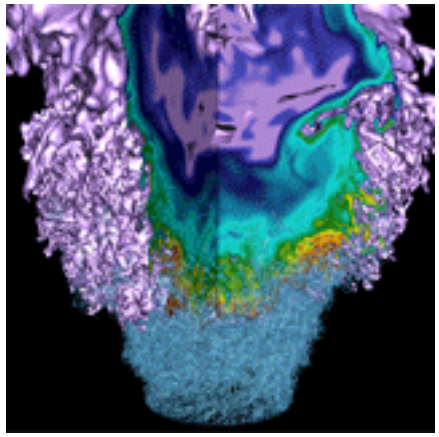
## Passwords

- Change it at https://nim.nersc.gov
- Answer security questions in NIM, then you can reset it yourself

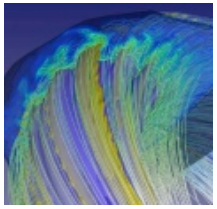
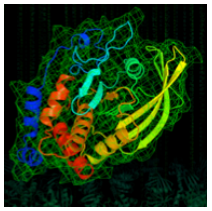
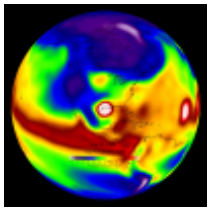
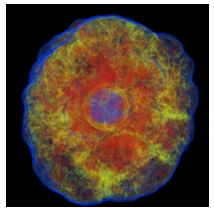
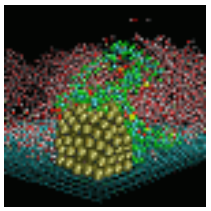
## Login Failures

- 5 or more consecutive login failures on a machine will disable your ability to log in
- Send e-mail to consult@nersc.gov to reset your failure count

# Data Resources



NERSC

# Data Storage Types

- **"Spinning Disk"**
  - Interactive access
  - I/O from compute jobs
  - "Home", "Project", "Scratch", "Projectb", ~~"Global Scratch"~~
  - Mendel and Mendel+ nodes have some local scratch space

- **Archival Storage**
  - Permanent, long-term storage
  - Tapes, fronted by disk cache
  - "HPSS" (High Performance Storage System)

# Home Directory

- When you log in you are in your "Home" directory.
- Permanent storage
  - Weeklong daily snapshots taken: $HOME/.snapshots
- The full UNIX pathname is stored in the environment variable $HOME

```
genepool04% echo $HOME
/global/homes/d/dudwary
```

- $HOME is a global file system
  - You see all the same directories and files when you log in to any NERSC computer.
- Your quota in $HOME is 40 GB and 1M inodes (files and directories).
- Use “myquota” command to check your usage and quota

# Scratch Directories

- "Scratch" file systems are large, high-performance file systems, intended to be temporary.
  - Standard projectb scratch size: 20TB and 4M inodes
- Significant I/O from your compute jobs should be directed to $SCRATCH
- Each Genepool user has a personal directory referenced by $SCRATCH and $BSCRATCH
  - on Genepool this points to /global/projectb/scratch/<username>
  - $SCRATCH is local on Edison and CORI (ie does not point to projectb).
- Data in $SCRATCH is purged (12 weeks from last access)
- Always save data you want to keep to HPSS (see below)
- Data in $SCRATCH is ***not*** backed up and could be lost if a file system fails.

# Project Directories

- All NERSC systems mount the NERSC global "Project" file systems.
- Projectb is specific to the JGI, but is also accessible on Edison and Cori.
- "Project directories" are created upon request for projects (groups of researchers) to store and share data.
- Data in /projectb/projectdirs is not purged. This may change in the future, but for long term storage, you should use the archive.

# IO Tips

- Use $SCRATCH for good IO performance
- Write large chunks of data (MBs or more) at a time
- Use a parallel IO library (e.g. HDF5)
- Read/write to as few files as practical from your code (try to avoid 1 file per MPI task)
- Use $HOME to compile unless you have too many source files or intermediate (*.o) files
- Do not put more than a few 1,000s of files in a single directory
- Save any and everything important to HPSS

# Archival Storage (HPSS)

- For permanent, archival storage
- Permanent storage is magnetic tape, disk cache is transient
  - 100PB data in >400M files written to 32k cartridges
  - Cartridges are loaded/unloaded into tape drives by sophisticated library robotics
- Front-ending the tape subsystem is 150TB fast-access disk

- **Hostname: archive.nersc.gov**
- **Over 100 Petabyes of data stored**
- **Data increasing by 1.7X per year**
- **150 TB disk cache**
- **8 STK robots**
- **44,000 tape slots**
- **Average data xfer rate: 100 MB/sec**

# Authentication

- **NERSC storage uses a token-based authentication method**
  - User places encrypted authentication token in ~/.netrc file at the top level of the home directory on the compute platform
- **Authentication tokens can be generated in 2 ways:**
  - Automatic – NERSC auth service:
    - Log into any NERSC compute platform; Type “hsi”; Enter NERSC password
  - Manual – https://nim.nersc.gov/ website
    - Under “Actions” dropdown, select “Generate HPSS Token”; Copy/paste content into ~/.netrc; chmod 600 ~/.netrc
- **Tokens are username and IP specific—must use NIM to generate a different token for use offsite**

# HPSS Clients

- **Parallel, threaded, high performance:**
  - HSI
    - Unix shell-like interface
  - HTAR
    - Like Unix tar, for aggregation of small files
  - PFTP
    - Parallel FTP
- **Non-parallel:**
  - FTP
    - Ubiquitous, many free scripting utilities
- **GridFTP interface (garchive)**
  - Connect to other grid-enabled storage systems

# Archive Technologies, Continued…

- **HPSS clients can emulate file system qualities**
  - FTP-like interfaces can be deceiving: the archive is backed by tape, robotics, and a single SQL database instance for metadata
  - Operations that would be slow on a file system, e.g. lots of random IO, can be impractical on the archive
  - It's important to know how to store and retrieve data efficiently. (See http://www.nersc.gov/users/training/nersc-training-events/data-transfer-and-archiving/)

- **HPSS does not stop you from making mistakes**
  - It is possible to store data in such a way as to make it difficult to retrieve
  - The archive has no batch system. Inefficient use affects others.

# Avoid Common Mistakes

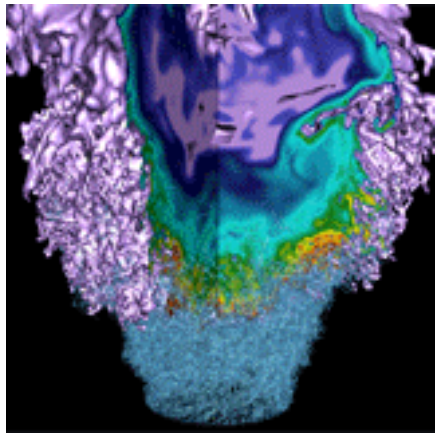
- **Don't store many small files**
  - Make a tar archive first, or use htar
- **Don't use to recursively store or retrieve large directory trees**
- **Don't stream data via UNIX pipes**
  - HPSS can't optimize transfers of unknown size
- **Don't pre-stage data to disk cache**
  - May evict efficiently stored existing cache data
- **Avoid directories with many files**
  - Stresses HPSS database
- **Long-running transfers**
  - Can be error-prone
  - Keep to under 24 hours
- **Use as few concurrent sessions as required**
  - Limit of 15 in place

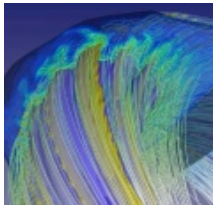
# Data Transfer and Archiving

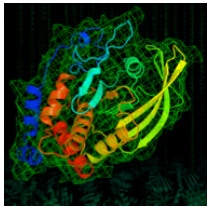
- **More detailed information:**

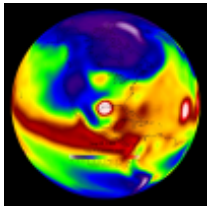
**http://www.nersc.gov/users/training/events/data-transfer-and-archiving/**

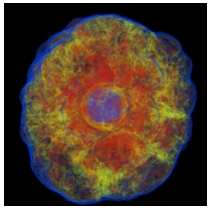
# Modules

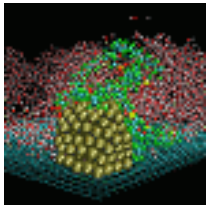










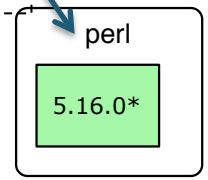


# Using Software and the UNIX Environment

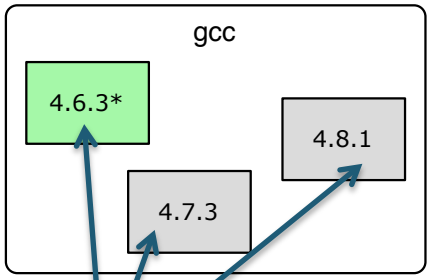
- **Providing large-scale installations of software for many different users on an HPC system presents a number of challenges:**
  - Different users need different software, use different shells
  - Some users need different specific versions, including older versions
  - All users need to access the software quickly and easily from "everywhere" [network-mounted, non-standard paths]
  - Providing a user interface for accessing that software can be challenging
    - Example: How would you use software installed in
      /usr/common/jgi/aligners/blast+/2.2.28
    - Answer:
      - Add /usr/common/jgi/aligners/blast+/2.2.28/bin to PATH;
      - csh: setenv PATH /usr/common/jgi/aligners/blast+/2.2.28/bin:$PATH
      - bash: export PATH=/usr/common/jgi/aligners/blast+/2.2.28/bin:$PATH

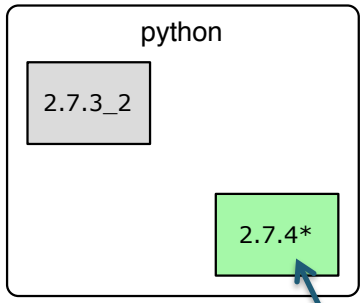
# What are Modules?

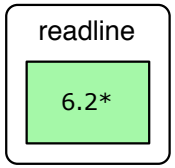
Modules have a name




A “module” is something that can be loaded or unloaded dynamically into the environment.




Modules have a version
can have *many versions*

Modules can have a *default* version

To refer to the *default version* of a module, use: <name>
e.g. module load gcc
To refer to a *specific version* of a module, use: <name>/<version>
e.g. module load gcc/4.8.1

# Basic Modules Functionality

- **Modules manipulate the environment**
  - Loading can:
    - Set an environment variable (possibly by replacing)
    - Append (or prepend) to a compound environment variable
    - Unset an environment variable
    - *can* execute a command (not recommended if the command changes the state of the system)
  - 'module unload' reverses the effects of the 'module load'
  - Which effects of a module might be irreversible?
    - Answer:
      - setenv won't restore the environment to its original state
      - multiple modules calling 'setenv' or 'unsetenv' on the same variable might lead to an inconsistent state (those modules should conflict)
      - Executing system calls which change system state (e.g. xhost) are not trivially reversible by unloading the module

# Modules: conflicting and swapping

- **Some modules are incompatible**
  - E.g. both wublast and blast+ provide different blastn, blastx, etc. executables
  - To prevent these modules from being simultaneously loaded, they conflict

```
dmj@genepool02:~$ module load wublast
dmj@genepool02:~$ module load blast+
blast+/2.2.26(25):ERROR:150: Module 'blast+/2.2.26' conflicts with the currently
loaded module(s) 'wublast/20060510'
```

- **Most of the time, only a single version of a module should be loaded at a time:**
  - e.g., doesn't make sense to load more than one version of gcc
  - Try:

```
module purge              ## cleans everything out
module load gcc
Module load gcc/4.8.1
```

  - Error? to change from gcc/4.6.3 (the default) to gcc/4.8.1 (the latest), swap!

```
module swap gcc gcc/4.8.1     -or-  module swap gcc/4.8.1
```

# Common Environment Variables in Modules

- **Modules for software packages commonly set:**
  - PATH
  - LD_LIBRARY_PATH
  - PYTHONPATH
  - PERL5DIR

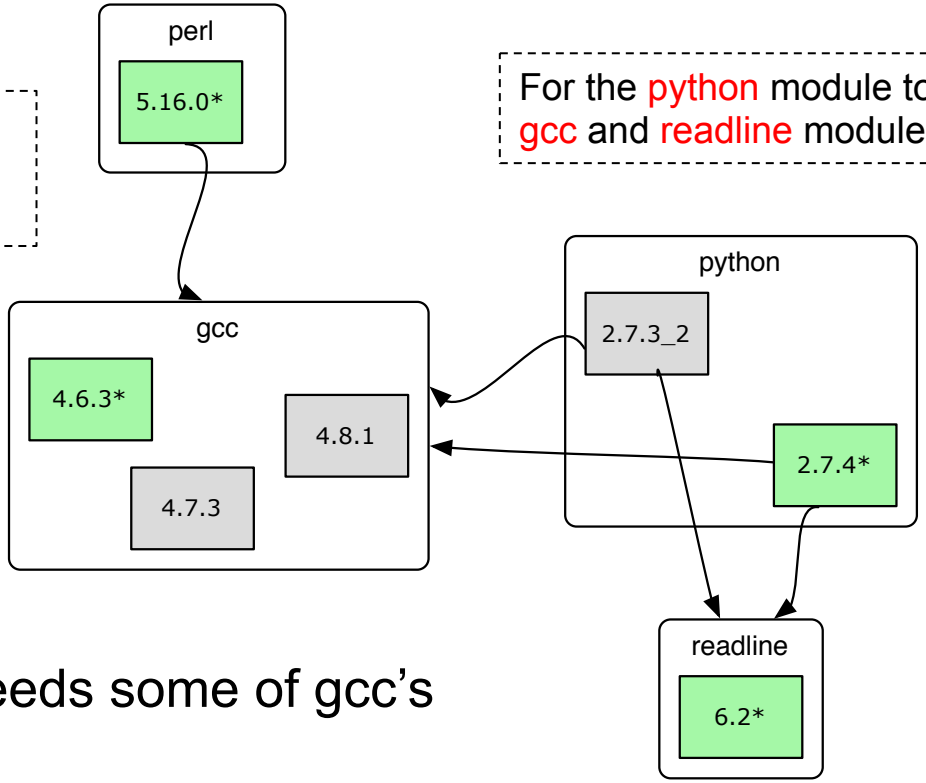
Be VERY careful about manipulating these environment variables!!!

- **Every usg/jgi module for software also sets an environment variable pointing to the base of the distribution:**
  - E.g. BOOST_ROOT, PERL_DIR, PYTHON_DIR, GIT_PATH
- **Exercise:**
  - Load the python module first
  - Use 'module info' to investigate the effects of:
    - graphviz
    - RSeQC
    - Smrtanalysis
  - Are there commonalities? Differences?

# Modules may have dependencies

For the perl module to function, the gcc module needs to be loaded

For the python module to function, *both* the gcc and readline modules need to be loaded




- Python needs some of gcc's libraries
- Perl needs some of gcc's libraries
- Python also needs readline's libraries

# Module commands reference

- **module list**
  - show all loaded modules
- **module avail <module name>**
  - list modules with <module name> that can be loaded
- **module load <module name>**
- **module unload**
- **module swap <current module> <new module>**
  - unload a loaded module and load the new one
- **module purge**
  - unload all modules (it's a good idea to start a batch script this way!)
- **module use <a directory>**
  - Use a different $MODULEPATH

**For Genepool-wide installation of new modules, or software upgrades, contact your consultants!**

# Using Modules in Batch Scripts

```
#!/bin/bash –l
#$ -l ram.c=10G
#$ -l h_rt=8:00:00

set –e

module purge
module load PrgEnv-gnu/4.6
module load python/2.7.4

module use /path/to/my/groups/modulefiles
module load MyPipeline/1.0

#…. Run your programs here ….
```

Ensures login environment is initialized

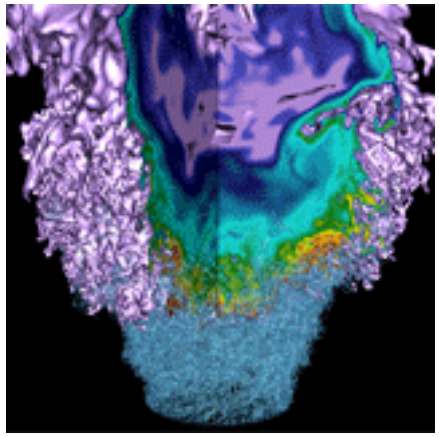
UGE options

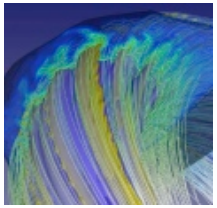
Kill script if any commands give non-zero exit status

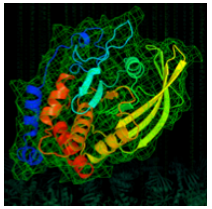
Clear all the modules, load any needed variant-provider modules

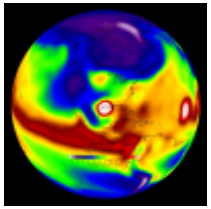
Add your modulefiles to MODULEPATH (module use)
Load your pipeline module

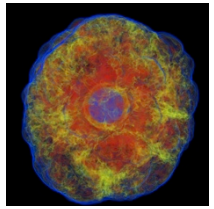
# Running and Monitoring Jobs

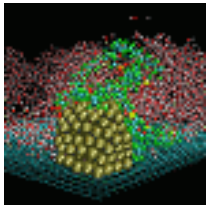












# Types of Jobs on genepool

- **Batch – Scheduled** **(compute nodes, fpga)**
  - 8,320 cores for 72,953,280 compute hours per year in genepool
  - use "qsub" to submit a job
- **Interactive – Scheduled** **(compute nodes subset)**
  - 80 cores presently, increasing size
  - use "qlogin" to submit a job
- **Interactive – Unscheduled** **(login nodes, gpints)**
  - 4 login nodes, 27 gpint nodes
  - ssh to the host, direct-use
- **Services – Unscheduled** **(login nodes, gpints, gpweb, gpdb, gpodb)**
  - Web services
  - Database services
  - Automated job submission / control

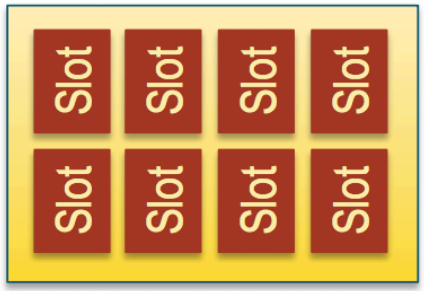
# Basics of Batch Jobs

- Genepool is a shared resource
- Each calculation usually only takes a small portion of genepool
  - Every job is strictly limited on the consumption of genepool resources
  - The job description specifies the resource limits
- Univa GridEngine is used to schedule each calculation on genepool
  - The scheduler matches job resource limit requests with physical resources

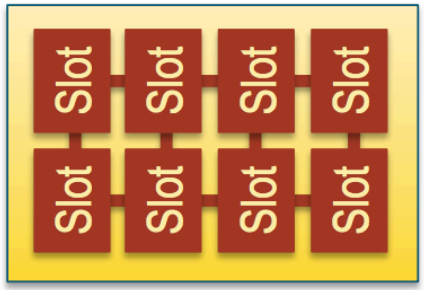
# Basics of GridEngine

- **GridEngine schedules "slots"**
  - Not memory, nor processors, nor nodes
- **A *slot* is a portion of a node**
  - For most nodes on genepool, a slot is defined as a single processor plus ($ram.c_{nodeTotal}/n_{cores}$) memory
  - Some nodes are *exclusively scheduled* – all slots on the node are bonded together as one schedulable unit
- **Jobs are placed in *queues***
  - Queues manage the resources of disparate sets of nodes, and have distinct resource limits
    - normal.q has a 12 hour time limit
    - long.q has a 10 day time limit
- **Jobs are scheduled in order of a balance of:**
  - Resource availability
  - Job prioritization

Node




Exclusive Node



# Basics of Batch Job Submission

## Example Batch Script

```
#!/bin/bash
module load blast+
input=$1
database=$2
blastn -query $input -db $database <more options>
```

## Submitting the example

```
genepool$ qsub -cwd example.sh queries.fa myDB
Your job 347283 ("example.sh") has been submitted.
```

- "qsub" submits the job for batch processing
- "-cwd" directs the job to work out of the present location in the filesystem
  - the current working directory
- Default resource limits will be applied, since none were specified
  - **1 slot**
  - **5.25GB memory/slot**
  - **12 hours**

# Many examples on the NERSC webpage

https://www.nersc.gov/users/computational-systems/genepool/running-jobs/submitting-jobs/

### qsub commands and options

UGE (Univa Grid Engine) is the batch system used for Genepool/Phoebe.

| Action | How to do it | Comment |
|---|---|---|
| Submit a job | qsub *script* | In UGE you need to submit a script, not an executable. |
| Specify number of processors for a threaded job | qsub -pe pe_slots 8 ... | Request 8 cores on a single node for your job. Please specify as many processors as will be needed during your job. |
| Specify number of nodes and processors for an MPI job | qsub -pe pe_8 16 ... | Request 2 nodes with 8 processors per node. pe_1, pe_2, pe_4, pe_8, pe_16, and pe_32 are available. |
| Specify memory required per processor | qsub -l ram.c=4G ... | Specify how much memory is required *per processor* for your job. At present this is implemented by implicity setting h_vmem (a virtual memory limit), so you will need to account for all virtual memory needed by your application. Use of a program like memtime during your benchmarking ahead of production may be informative. |
| Specify a time limit for your job | qsub -l h_rt=6:00:00 ... | Specifies that your job will run for at most 6 hours. Default is 12 hours. If you request more than 12 hours, your job will enter the long queue, which has much fewer dedicated resources. |
| Submit a job to the high priority queue | qsub -l high.c *script* | The high.c complex is for small fast turn around jobs |
| Submit a job that depends on other jobs | qsub -hold_jid [job_ID\|job_name] *script* | UGE just recognizes whether or not [job_ID\|job_name] is finished before submitting your job. The newly submitted job will only start once all jobs in the hold_jid list are completed. |
| Submit a job to different project | qsub -P [project] *script* | By default your job runs as the project corresponding to your primary NERSC project repo. If qsub indicates you do not have access to the project you specify please file a ticket to get added to it. |
| Get e-mail from your job upon completion | qsub -m e -M <email address> ... | No email by default. UGE can also email at the beginning of a job with "-m b", or upon errors with "-m a". |

# Genepool Queues

Exclusive = all CPUs on a node

| Queue Name | Purpose | User Requestable | Slot Limit | Memory Limit | Wall Clock Limit |
|---|---|---|---|---|---|
| **normal.q** | Production workloads. Default queue | No, queue assignment based on resource requests (ram.c, h_rt) | none | 42G | 12 hours |
| **long.q** | Production workflows that need more than 12 hours | No, queue assignment based on resource requests (ram.c, h_rt) | 320 | 42G | 240 hours |
| **normal_excl.q** | Production workloads - exclusive node scheduling. | No, queue assignment based on resource requests (ram.c, h_rt, exclusive.c) | none | >42G | 12 hours |
| **long_excl.q** | Production workloads that require more than 12 hours. | No, queue assignment based on resource requests (ram.c, h_rt, exclusive.c) | none | >42G | 240 hours |
| **high.q** | High priority jobs and debugging jobs | Yes, request either "-q high.q" or "-l high.c" (deprecated) | 8 | 120GB | 240 hours<br>default 12 hours |
| **interactive.q** | For light-weight interactive jobs; default for qlogin | Only with qlogin (default) or special services | none | 120GB | 240 hours |
| **timelogic.q** | Access to Timelogic accelerated blast nodes | Yes, request either "-q timelogic.q" or "-l timelogic.c" (deprecated) | none | 800MB | none |
| **xfer.q** | Data Transfer Queue on genepool; Use this to transfer data to /global/dna | Yes, request "-l xfer.c". | 2 | 3.25GB | 72 hours |

# Pointers to avoid common mistakes

- **Be aware of how many threads your software will use, and be sure you've requested the right number (typically with "qsub –pe pe_slots 8")**
  - hmmer is commonly problematic – by default will try to take all CPUs on a machine, unless otherwise specified
- **If at all possible, use 12 hours or less**
  - The long queue has few nodes, and usage is constrained
- **Use –cwd or –wd <directory> with qsub**
  - Writing output to your home directory can slow everyone down. Write to scratch!

# qstat

```
dmj@phoebe:~$ qstat
job-ID  prior   name       user         state submit/start at     queue                          jclass                         slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------------------------------------
 336024 0.44577 testJob_1  dmj          r     02/11/2013 19:30:03 normal.q@sgi07a26.nersc.gov                                    1
 336025 0.39718 testJob_2  dmj          r     02/11/2013 19:30:03 normal.q@sgi07b08.nersc.gov                                    1
 336026 0.37289 testJob_3  dmj          r     02/11/2013 19:30:03 normal.q@sgi07b13.nersc.gov                                    1
 336027 0.00000 env        dmj          qw    02/11/2013 19:30:08                                                                1
dmj@phoebe:~$
```

- **By default, qstat only shows *your* jobs**
- **To see others, qstat -u <username> or qstat -u ***
- **State:**
  - r: “running”
  - qw: “queue-wait”
  - R<state>: “rescheduled <basic state>”
  - E<state>: “error <basic state>”
  - h<state>: “hold <basic state>”

# Investigating Completed Jobs

- **GridEngine saves accounting information for all completed and errored-out jobs**
- **These records reflect what your project has been billed for fair-share calculations**
- **Also show the total resource utilization figures**
  - Can be useful (but not perfect) when trying to understand why a job crashed

# Investigating Completed Jobs

- **Check your jobs for the past 90 days:**
  - qqacct -D 90 -q 'user=="dmj"'
- **Just the jobs UGE thinks failed over past 3 days (default)**
  - qqacct -q 'user=="dmj" && failed != 0'
- **Just the jobs UGE thinks failed with time/memory info**
  - qqacct -q 'user=="dmj" && failed !=0' -c 'job_number,failed,memory(ppn*h_vmem),memory(maxvmem), h_rt,wall'
- **Always put query in single quotes – the shell is likely to try to parse many of the characters in the query**
- **"-c" overrides default output columns**


NERSC